

Grupo Handbook

Handbook de Questões de TI

comentadas para **CONCURSOS**

Além do gabarito

2ª Edição

Volume 2

*Analista de Sistemas - Eng. de Software
Petrobras 2008
Fundação Cesgranrio*

Prefácio

Este é o segundo volume da série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito*. Ele traz a prova para Analista de Sistemas Júnior - Engenharia de Software, aplicada em junho de 2008 pela Fundação Cesgranrio. São 50 questões comentadas “além do gabarito” para você se preparar não só para os concursos para a Petrobras, mas, também, para todos os demais concursos de alta concorrência na área de TI.

Estamos vivendo a era do petróleo no Brasil. A Petrobras está aproveitando muito bem essa fase para maximizar a sua produção e, por consequência, os seus lucros. Nessa jornada, surge a necessidade de novos profissionais qualificados, inclusive na área de TI. Para suprir essa necessidade, a empresa vem realizando concursos com maior frequência.

Alguns fatores fazem com que concursos para a Petrobras sejam altamente concorridos, tais como: contato com diversas tecnologias de ponta; excelente ambiente de trabalho; salário e benefícios superiores aos da maioria das empresas e órgãos governamentais. Tendo em vista essa alta concorrência, é de fundamental importância que os materiais de estudo do concurseiro sejam de ótima qualidade.

Este volume traz questões reais que abordam temas como segurança de informação, arquitetura de computadores, banco de dados e governança de TI. Não faltará embasamento teórico ao concurseiro, uma vez que os comentários elaborados não se limitam à simples resolução das questões.

Bons estudos,

Grupo Handbook de TI

Direitos Autorais

Este material é registrado no Escritório de Direitos Autorais (EDA) da Fundação Biblioteca Nacional. Todos os direitos autorais referentes a esta obra são reservados exclusivamente aos seus autores.

Os autores deste material não proíbem seu compartilhamento entre amigos e colegas próximos de estudo. Contudo, a reprodução, parcial ou integral, e a disseminação deste material de forma indiscriminada através de qualquer meio, inclusive na Internet, extrapolam os limites da colaboração. Essa prática desincentiva o lançamento de novos produtos e enfraquece a comunidade concurseira Handbook de TI.

A série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito* é uma produção independente e contamos com você para mantê-la sempre viva.

Grupo Handbook de TI

Canais de Comunicação

A equipe Handbook de TI disponibiliza diversos canais de comunicação para seus clientes.

Loja Handbook de TI

<http://www.handbookdeti.com.br>

Serviço de Atendimento

Comunicação direta com a Equipe Handbook de TI pode ser feita em
<http://www.handbookdeti.com.br/contacts>

Twitter do Handbook de TI

Que acompanhar de perto o trabalho do Grupo Handbook de TI. Cadastre-se no twitter e comece a seguir o grupo Handbook de TI em <http://twitter.com/handbookdeti>

1. **Assuntos relacionados:** *Arquitetura de Computadores, Modos de Endereçamento de Memória,***Banca:** CESGRANRIO**Instituição:** Petrobras**Cargo:** Analista de Sistemas - Eng. de Software**Ano:** 2008**Questão:** 21

Um computador tem um registrador R e um conjunto de instruções de um operando, todas com modo de endereçamento indireto. Três destas instruções são especificadas a seguir.

LD: Copia da memória principal para o registrador R.

AC: Adiciona da memória principal ao registrador R.

ST: Move do registrador R para a memória principal.

Considere o programa apresentado abaixo, executado no computador, acessando o bloco de memória principal, cuja situação inicial é mostrada a seguir.

Endereço	Valor Armazenado
00H	01H
01H	02H
02H	03H
03H	04H
04H	05H

LD 01H

AC 02H

ST 03H

AC 00H

ST 01H

LD 03H

ST 00H

Considere que tanto o endereçamento quanto os valores envolvidos nas operações utilizam apenas um byte de memória cada. Após a execução do programa, qual será, em hexadecimal, a soma dos valores armazenados no bloco de memória?

(a). 00H

(b). 04H

(c). 0AH

(d). 10H

(e). 1CH

Solução:

Primeiramente, os conceitos de endereçamento de dados devem estar bem claros. Em uma instrução de programa, há várias maneiras de referenciar um valor, as mais conhecidas são:

- *Imediato*: o valor do operando é especificado diretamente na instrução. Sua principal vantagem é não requerer acesso à memória para obter o operando. A desvantagem é que esse modo impõe uma limitação no tamanho do operando. Suponha que o computador

descrito suporte acesso imediato. A instrução LD 30H faria com que o valor 30H fosse copiado para o registrador R. Entretanto, há ocasiões em que não somente um byte deve ser copiado, por exemplo LD 201040H. Nesse caso, como o valor é armazenado diretamente na instrução, seria necessário aumentar o tamanho da instrução e isso não é possível na maioria das arquiteturas de computador;

- *Direto*: o campo de endereço contém o endereço efetivo do operando na memória. Requer, portanto, apenas um acesso para determinar o valor do operando. Sua limitação é fornecer um espaço de endereçamento limitado. Suponha que o computador descrito suporte endereçamento direto. A instrução LD 01H, faria com que o valor armazenado na posição de memória 01H, ou seja, 02H fosse copiado. Entretanto, se a instrução possuir somente um byte para o endereçamento direto, por exemplo, a quantidade de posições de memória estará limitada em 256 (2^8);
- *Indireto*: o campo de endereço aponta para uma posição de memória que contém o endereço de memória do operando. Sua principal desvantagem é a necessidade de dois acessos à memória. A vantagem em relação ao modo de endereçamento direto é o aumento do espaço de endereçamento, que passa a ser igual 2^n , onde n é o tamanho da palavra na memória. Suponha que o computador tenha somente um byte para endereçar a posição de memória, mas que essa posição de memória corresponda a uma palavra com tamanho de 2 bytes. Um endereçamento na forma indireta, possibilitará o endereçamento de 65536 posições de memória (2^{16}) e não mais 256 como no endereçamento direto. Não é o caso da questão, onde tanto o tamanho permitido para endereçamento na instrução e o tamanho da palavra de memória são iguais a um byte;
- *Registrador*: é semelhante ao modo direto, no entanto, o campo de endereço se refere a um registrador e não a uma posição de memória. Geralmente, esse campo é composto por 3 ou 4 bits, o que permite referenciar de 8 a 16 registradores de propósito geral. Suas vantagens são o tamanho pequeno do campo de endereço e a não necessidade de se acessar à memória. Sua desvantagem é o espaço de endereçamento limitado pelo número de registradores. Por exemplo, poderíamos supor que o computador da questão permitisse endereçamento por registrador e tivesse 16 registradores. Assim, seria possível que um registrador além do R, por exemplo S, pudesse ser endereçado como 05H. Uma instrução da maneira LD 05H copiaria o valor do registrador S para o registrador R;
- *Indireto via Registrador*: semelhante ao modo de endereçamento indireto. O campo de endereço aponta para o registrador que contém a posição de memória do operando. Sua vantagem é a necessidade de um único acesso à memória, um a menos que no modo indireto;
- *Deslocamento*: requer que uma instrução tenha dois campos de endereço, com pelo menos um explícito. O valor de um dos campos é usado diretamente (valor = A). O outro campo é baseado no código da operação, e especifica um registrador cujo conteúdo é adicionado à A, para produzir o endereço efetivo. Os três modos de endereçamento por deslocamento são: *relativo*, *via registrador-base* e *indexado*;
- *Pilha*: a pilha é um bloco reservado de posições de memória. Elementos podem ser colocados e removidos do topo da pilha. O apontador do topo da pilha (*stack-pointer*) é mantido em um registrador. Portanto, de fato, referências a pilha são feitas por endereçamento indireto via registrador.

Já que a questão trata de endereçamento indireto, o valor armazenado no local especificado pelo operando é o endereço de memória do valor que será utilizado na operação. Por exemplo, a instrução LD 01H, carrega, no registrador R, o valor 03H, pois no endereço 01H está

armazenado o endereço 02H, que por sua vez, contém o valor desejado, 03H. Seguindo os passos do programa, teremos:

1. LD 01H, R \leftarrow 03H, R recebe o valor armazenado no endereço 02H;
2. AC 02H, R \leftarrow 03H + 04H \leftarrow 07H, o valor de R é somado ao valor armazenado no endereço 03H;
3. ST 03H, [04H] \leftarrow 07H, a posição de memória 04H recebe o valor do registrador R;
4. AC 00H, R \leftarrow 07H + 02H \leftarrow 09H, o valor de R é somado ao valor da posição 01H;
5. ST 01H, [02H] \leftarrow 09H, a posição de memória 02H recebe o valor de R;
6. LD 03H, R \leftarrow 07H, R recebe o valor armazenado no endereço 04H;
7. ST 00H, [01H] \leftarrow 07H, a posição de memória 01H recebe o valor de R.

Após o término do programa, a situação final do bloco de memória será de acordo com a Tabela 1.

Tabela 1: situação final do bloco de memória.

Endereço	Valor Armazenado
00H	01H
01H	07H
02H	09H
03H	04H
04H	07H

A soma é $01H + 07H + 09H + 04H + 07H = 1CH$, que, em decimal, é 28. Logo, a alternativa correta é a letra (e).

2. Assuntos relacionados: *Arquitetura de Computadores, Thread,***Banca:** *CESGRANRIO***Instituição:** *Petrobras***Cargo:** *Analista de Sistemas - Eng. de Software***Ano:** *2008***Questão:** *22*

Alguns sistemas operacionais permitem que seus processos criem múltiplos threads de execução. Em operação normal, o que é previsto que os threads de um mesmo processo do sistema operacional compartilhem?

- (a). Arquivos abertos
- (b). Registradores
- (c). Pilha (stack)
- (d). Variáveis locais de cada thread
- (e). Contador de instrução (program counter)

Solução:

Uma thread é comumente definida como um fluxo único de controle sequencial dentro de um programa. O uso de threads visa reduzir o custo do gerenciamento de processos, que consiste principalmente em:

- criação do processo;
- trocas de contextos entre processos;
- overhead associado a esquemas de proteção de memória;
- comunicação entre processos.

Podemos dizer que as threads pertencentes ao mesmo processo utilizam os recursos alocados no sistema operacional para esse processo, como:

- o espaço de endereçamento na memória;
- os arquivos abertos (handles);
- os objetos de sincronização.

O compartilhamento desses recursos permite que os fluxos de execução (threads) se comuniquem eficientemente. Então, a letra (a) é a opção correta.

Entretanto, threads dentro de um processo (e também entre processos) são escalonadas e executadas independentemente. No momento de sua execução, cada thread recebe alguns recursos próprios, como:

- os registradores;
- a pilha de execução, que lhe dará poder para chamar métodos, passar parâmetros e alocar variáveis locais;
- o contador de instrução (program counter), que é essencial para que o fluxo de execução prossiga.

É importante perceber que para que o acesso a esses recursos exclusivos ocorra, é necessário o chaveamento de contexto entre as threads, ou seja, o estado dos elementos próprios citados deverá ser armazenado e restaurado a cada troca de thread no uso do processador.

Dado o exposto, as alternativas (b), (c), (d) e (e) podem ser eliminadas, pois citam recursos que não são compartilhados entre threads. A Figura 1 exemplifica o compartilhamento das threads dentro de um processo.

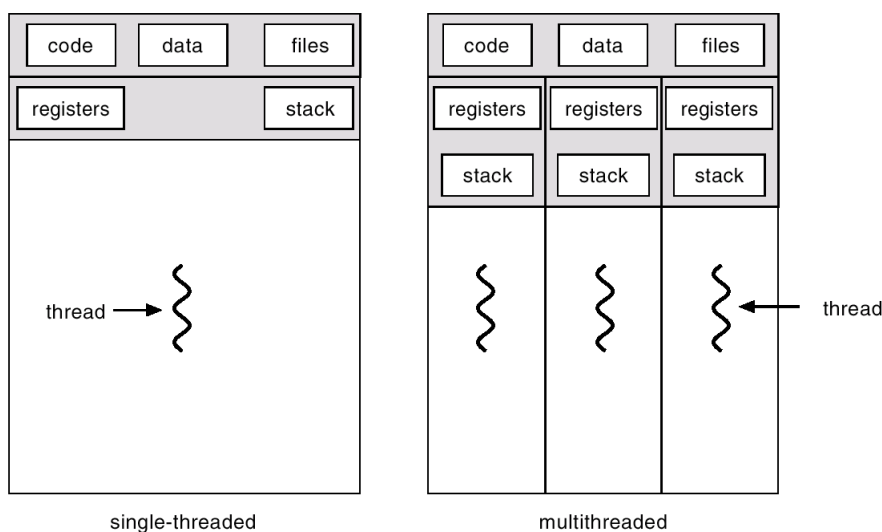


Figura 1: exemplificação de um processo com uma única thread e com múltiplas threads.

Concluindo, as principais vantagens do uso de threads são:

- permite a exploração do paralelismo real oferecido por máquinas multiprocessadas;
- possibilita o aumento do número de atividades executadas por unidade de tempo (throughput);
- permite sobrepor operações de cálculo com operações de I/O e, com isso, reduzir o tempo de resposta;
- o tempo de criação e destruição de threads é inferior ao tempo de criação e destruição de processos, respectivamente;
- o chaveamento de contexto entre threads é mais rápido que o tempo de chaveamento entre processos;
- como threads compartilham o descritor do processo, elas dividem o mesmo espaço de endereçamento, o que permite a comunicação por memória compartilhada sem interação com o núcleo (kernel) do sistema operacional.

3. **Assuntos relacionados:** *Banco de Dados, Modelo Relacional, Normalização de Banco de Dados, Primeira Forma Normal (1FN),*

Banca: CESGRANRIO

Instituição: Petrobras

Cargo: Analista de Sistemas - Eng. de Software

Ano: 2008

Questão: 23

É correto afirmar que qualquer relação válida de um modelo relacional:

- (a). pode apresentar tuplas duplicadas, desde que não haja chaves candidatas definidas.
- (b). em seus atributos ordenados da esquerda para a direita, de acordo com a definição.
- (c). tem suas tuplas naturalmente ordenadas, para fins de localização.
- (d). tem um índice físico para cada chave candidata, incluindo a chave primária.
- (e). está, pelo menos, na primeira forma normal.

Solução:

Conceitualmente, em um banco de dados relacional, as relações podem ser definidas como um conjunto de tuplas. Uma tupla é uma sequência ordenada de atributos e representa, usualmente, um objeto do mundo real e suas informações. Todas as tuplas em uma mesma relação possuem o mesmo conjunto de atributos. Entretanto, em uma relação, elas devem preservar o que é chamado de integridade existencial, que, basicamente, implica que tuplas iguais (com todos os valores dos seus atributos iguais) não são permitidas. Caso contrário, não temos uma relação.

A forma de garantir a integridade existencial é definir uma chave primária, que, obrigatoriamente, deve ser não-nula e única em toda relação. Logicamente, sendo a chave primária única, as tuplas duplicadas não serão permitidas e a relação estará garantida.

Deve ficar clara a diferença entre tabela e relação. Uma tabela nada mais é do que um conjunto de linhas e colunas. Já as relações, que já foram definidas acima, são implementadas fisicamente em tabelas, que, obviamente, devem atender às condições exigidas em uma relação. Ou seja, nem toda tabela representa uma relação.

Na definição da primeira forma normal dada por Date, uma tabela está na primeira forma normal se, e somente se, for isomórfica à alguma relação. Isso quer dizer que, especificamente, a tabela deve atender às cinco condições abaixo:

- não existe uma ordenação das linhas de cima para baixo;
- não existe uma ordenação das colunas da direita para esquerda;
- não existem linhas duplicadas;
- qualquer interseção linha-coluna deve conter exatamente um valor no domínio aplicável e nada mais;
- todas as colunas são regulares, no sentido de não possuírem componentes ocultos como identificadores de linhas, identificadores de colunas, identificadores de objetos ou timestamps ocultos. Veja nas Tabelas 2, 3 e 4 um exemplo de cenário em que a informação contida em uma tabela que não está na 1FN é transportada para outras tabelas que estão na 1FN.

idPessoa	nmPessoa	dtAniversario	nrTelefones
1	João Roberto	01/05/1980	9311-9654 - 3698-5741
2	Juliana Gomes	28/02/1985	3232-4521 - 6352-9821 - 3987-8855
3	Talita Brandão	03/12/1988	5561-9874

Tabela 2: exemplo de tabela que não está na 1FN.

idPessoa	nmPessoa	dtAniversario
1	João Roberto	01/05/1980
2	Juliana Gomes	28/02/1985
3	Talita Brandão	03/12/1988

Tabela 3: exemplo de tabela que está na 1FN.

idTelefone	idPessoa	nrTelefone
1	1	9311-9654
2	1	3698-5741
3	2	3232-4521
4	2	6352-9821
5	2	3987-8855
6	3	5561-9874

Tabela 4: exemplo de tabela que está na 1FN.

Dada a exposição teórica, podemos analisar as alternativas da questão:

- falsa, uma relação válida não permite tuplas duplicadas.
- falsa, a ordenação dos atributos não define a relação.
- falsa, a ordenação das linhas não define a relação.
- falsa, a relação é uma definição no nível conceitual. Os índices físicos são definidos no nível físico e não influenciam no conceito de relação. Geralmente, esses índices podem ser livremente criados para qualquer conjunto de atributos, já a obrigação citada de serem criados para cada chave candidata não existe e não faz nenhum sentido.
- verdadeira, pois, por definição, uma tabela representa uma relação se, e somente se, a tabela está na primeira forma normal.